# Fast Newton Nearest Neighbors Boosting For Image Classification

Wafa Bel Haj Ali, Richard Nock, Franck Nielsen, Michel Barlaud

2013 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 22–25, 2013, SOUTHAPMTON, UK

# FAST NEWTON NEAREST NEIGHBORS BOOSTING FOR IMAGE CLASSIFICATION

*Wafa Bel haj ali* *Richard Nock* *Frank Nielsen* *Michel Barlaud*

I3S / CNRS - U. Nice
Sophia Antipolis, France
belhajal@i3s.unice.fr

U. Antilles-Guyane
Schoelcher, Martinique
rnock@martinique.univ-ag.fr

Sony CSL
Tokyo, Japan
nielsen@lix.polytechnique.fr

I3S / CNRS - U. Nice
Sophia Antipolis, France
barlaud@i3s.unice.fr

## ABSTRACT

Recent works display that large scale image classification problems rule out computationally demanding methods. On such problems, simple approaches like $k$-NN are affordable contenders, with still room space for statistical improvements under the algorithmic constraints. A recent work showed how to leverage $k$-NN to yield a formal boosting algorithm. This method, however, has numerical issues that make it not suited for large scale problems.

We propose here an Adaptive Newton-Raphson scheme to leverage $k$-NN, $\mathrm{N}^3$, which does not suffer these issues. We show that it is a boosting algorithm, with several key algorithmic and statistical properties. In particular, it may be sufficient to boost a *subsample* to reach desired bounds for the loss at hand in the boosting framework. Experiments are provided on the SUN, and Caltech databases. They confirm that boosting a subsample — sometimes containing few examples only — is sufficient to reach the convergence regime of $\mathrm{N}^3$. Under such conditions, $\mathrm{N}^3$ challenges the accuracy of contenders with lower computational cost and lower memory requirement.

***Index Terms***— Machine learning

## 1. INTRODUCTION

Large scale image classification implies satisfying tight time, memory and numerical processing requirements. Coping with them involves in general two kinds of approaches. For the first one, scalability goes hand in hand with simplification: algorithms are built around sophisticated, state-of-the art approaches that are simplified to fit into these requirements, such as Support Vector Machines (SVM) with linear kernels [1], or (Ada)Boosting with weight clipping and simple stumps as weak classifiers [2].

The second kind of approaches use as core very simple algorithms that already fit into these requirements, and then, from this basis, elaborate more complex approaches with improved performances: this is the case for the $k$-nearest neighbor (NN) classifier, or the nearest class mean classifier embedded with metric learning [3, 4]. From the experimental standpoint, these latter approaches obtain surprising competitive results with respect to the former ones. In fact, they may have another advantage: while theoretical guarantees barely survive extreme simplification, elaborating on a core makes it perhaps easier to preserve its theoretical properties, such as its statistical consistency (*e.g.* for $k$-NN [5]).

Our paper belongs to the second category of approaches, as we elaborate on the ordinary $k$-NN classifier. Our approach is different but complementary to metric learning approaches, as we choose to adapt $k$-NN to the boosting framework.

One recent approach exists in this line of works [6], but it is not of Newton-Raphson type, and the numerical constraints for the computations of the weights updates and the leveraging coefficients make it impracticable for large scale classification.

Our high-level contribution is threefold:
$(i)$ a proof of the boosting ability of $\mathrm{N}^3$, the first boosting-compliant convergence rates for a Newton-type approach to convex loss minimization to the best of our knowledge;
$(ii)$ a divide and conquer algorithm to compute these estimators and cope with the curse of dimensionality with low memory requirement;
$(iii)$ experimentally optimized core-processing stages for $\mathrm{N}^3$ with linear cost per boosting iteration. Experimental results display that $\mathrm{N}^3$ manages to challenge accuracy of sophisticated approaches while being faster, and requires low memory.

The remaining of the paper is organized as follows: Section 2 states basic definitions. Section 3 presents classification-calibrated losses. Section 4 presents $\mathrm{N}^3$. Sections 5 discuss its theoretical properties. Section 6 presents experiments, and section 7 concludes the paper.

## 2. PROBLEM STATEMENT

We first provide some basic definitions. Our setting is multiclass, multilabel classification. We have access to an input set of $m$ examples (or prototypes), $\mathcal{S} \doteq \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m\}$. Vector $\boldsymbol{y}_i \in \{-1, +1\}^C$ encodes class memberships, assuming $y_{ic} = +1$ means that observation $\boldsymbol{x}_i$ belongs to class $c$. A classifier $H$ is a function mapping observations

978-1-4799-1180-6/13/$31.00 

| crit | transfer function $f$ | calibrated loss $F$ |
|---|---|---|
| A | $\frac{1}{1+\exp(-x)}$ | $\ln(1+\exp(-x))$ |
| B | $\frac{1}{1+2^{-x}}$ | $\ln(1+2^{-x})$ |
| C | $\frac{1}{2}\left(1+\frac{x}{\sqrt{1+x^2}}\right)$ | $\exp\sinh^{-1}(-x)$ |
| D | $\frac{1+\mathrm{H}(-x)}{2+\lvert x\rvert}$ | $\mathrm{H}(x)-\ln(2+\lvert x\rvert)$ |

**Table 1**. Calibrated losses that match (3) for several transfer functions. From top to bottom, losses are the logistic loss, binary logistic loss, Matsushita's loss, calibrated Hinge loss.

to vectors in $\mathbb{R}^C$. Given some observation $\boldsymbol{x}$, the sign of coordinate $c$ in $H(\boldsymbol{x})$ gives whether $H$ predicts that $\boldsymbol{x}$ belongs to class $c$, while its absolute value may be viewed as a confidence in classification.

The nearest neighbors (NNs) rule belongs to the oldest, simplest and still most widely studied classification algorithms [5]. It relies on a non-negative real-valued "distance" function. This function measures how much two observations differ from each other, and may not necessarily satisfy the requirements of metrics. We let $j \rightarrow_k \boldsymbol{x}$ denote the assertion that example $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, or simply example $j$, belongs to the $k$ NNs of observation $\boldsymbol{x}$. We shall abbreviate $j \rightarrow_k \boldsymbol{x}_i$ by $j \rightarrow_k i$ — in this case, we say that example $i$ belongs to the *inverse neighborhood* of example $j$. To classify an observation $\boldsymbol{x}$, the $k$-NN rule $H(\boldsymbol{x})$ computes the sum of class vectors of its nearest neighbors, that is: $H_c(\boldsymbol{x}) \doteq \sum_{j \rightarrow_k \boldsymbol{x}} y_{jc}$ is the coordinate $c$ in $H(\boldsymbol{x})$. A *leveraged* $k$-NN rule [6] generalizes this to:

$$H_c(\boldsymbol{x}) \doteq \sum_{j \rightarrow_k \boldsymbol{x}} \alpha_{jc} y_{jc} \ , \tag{1}$$

where $\boldsymbol{\alpha}_j \in \mathbb{R}^C$ leverages the classes of example $j$. Leveraging nearest neighbors raises the question as to whether there exists efficient inductive learning schemes for these *leveraging coefficients*.

To learn them, we adopt the framework of [7, 8], and focus on the minimization of a *total calibrated risk* which sums per-class losses:

$$\varepsilon_F(H, \mathcal{S}) \doteq \frac{1}{C}\sum_{c=1}^{C} \underbrace{\frac{1}{m}\sum_{i=1}^{m} F(y_{ic} H_c(\boldsymbol{x}_i))}_{\varepsilon_F(H_c, \mathcal{S})} \ . \tag{2}$$

To be classification-calibrated, loss $F : \mathbb{R} \rightarrow \mathbb{R}$ is required to be convex, differentiable and such that $F'(0) < 0$ [7] (Theorem 4), [8]. The recent advances in the understanding and formalization of (multiclass) loss functions suitable for classification have essentially concluded that classification calibration is mandatory for the loss to be Fisher consistent or proper [7, 8]. These are crucial properties without which the minimization of the loss brings no string statistical guarantee with respect to Bayes rule (such as universal consistency).

## 3. CLASSIFICATION-CALIBRATED LOSSES

In this paper, we are interested in a subset of classification-calibrated functions, namely those for which:

$$F(x) \doteq -x + \int f \ , \tag{3}$$

for some continuous *transfer function* $f : \mathbb{R} \rightarrow [0, 1]$, increasing and symmetric with respect to $(0, 1/2 = f(0))$. Intuitively, a transfer function brings an estimate of posteriors: it is a bijective mapping between a real-valued prediction $H_c(\boldsymbol{x})$ and a corresponding posterior estimation for the class, $\hat{p}[y_c = +1|\boldsymbol{x}]$, mapping which states that both values are positively correlated, and establishes a tie for $H_c = 0$ to which corresponds $\hat{p}[y_c = +1|\boldsymbol{x}] = 1/2$. Transfer functions have a longstanding history in optimization [9], and the set of $F$ that match (3) strictly contains balanced convex losses, functions with appealing statistical properties [6] (and references therein). Table 1 provides four example of such losses on which we focus. The calibrated Hinge loss relies on the linear Hinge loss:

$$\mathrm{H}(x) \doteq \max\{0, -x\} \ . \tag{4}$$

Another example of losses that meet (3) is the squared loss, for transfer $f = \min\{1, \max\{0, x + 1/2\}\}$.

To carry out the minimization of (2), we adopt a mainstream 1-vs-rest boosting scheme which, for each $c = 1, 2, ..., C$, carries out separately the minimization of $\varepsilon_F(H_c, \mathcal{S})$ in $\varepsilon_F(H, \mathcal{S})$. To do so, it fits the $c^{th}$ coordinate in leveraging coefficients by considering the two-class problem of class $c$ versus all others.

## 4. N$^3$: ADAPTIVE NEWTON NEAREST NEIGHBORS

### 4.1. Algorithm

We now present algorithm N$^3$, which stands for "Newton Nearest Neighbors". N$^3$ updates iteratively the leveraging coefficients of an example in $\mathcal{S}$, example picked by an oracle, WEO for "Weak Example Oracle". We detail below the properties and implementation of WEO. The technical details of the N$^3$ are given in Table 2. N$^3$ follows the boosting scheme, with iterative updates of leveraging coefficients followed by an iterative re-weighting of examples. Before embarking into formal algorithmic and statistical properties for N$^3$, we first show that N$^3$ is of Newton-Raphson type.

**Theorem 1** N$^3$ *performs adaptive Newton-Raphson steps to minimize* $\varepsilon_F(H_c, \mathcal{S})$, $\forall c$.

**Algorithm 1:** Algorithm NEWTON NN, $\mathrm{N}^3(\mathcal{S}, \mathrm{crit}, k)$

**Input**: Sample $\mathcal{S}$, criterion $\mathrm{crit} \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$, $k \in \mathbb{N}_*$;
Let $\boldsymbol{\alpha}_j \leftarrow \mathbf{0}, \forall j = 1, 2, ..., m$;
**for** $c = 1, 2, ..., C$ **do**
  //Minimize $\varepsilon_F(H_c, \mathcal{S})$
  Let $w_i \leftarrow \frac{1}{\|\mathbf{1} + y_{ic}\boldsymbol{y}_i\|_1}$, $\forall i$;
  **for** $t = 1, 2, ..., T$ **do**
    **[I.0]**//Choice of the example to leverage
    Let $j \leftarrow \textsc{Weo}(\mathcal{S}, \boldsymbol{w})$;
    **[I.1]**//Leveraging update, $\delta_j$
    Let $\eta(c, j) \leftarrow \sum_{i:j \rightarrow_k i} w_{ti} y_{ic} y_{jc}$;
    Let $n_j \leftarrow |\{i : j \rightarrow_k i\}|$;
    Compute $\delta_j$ following Table 2, using crit;
    **[I.2]**//Weights update
    $\forall i : j \rightarrow_k i$, update $w_i$ as in Table 2, using crit;
    **[I.3]**//Leveraging coefficient update
    Let $\alpha_{jc} \leftarrow \alpha_{jc} + \delta_j$;

**Output**: $\mathcal{H}(\boldsymbol{x}) \doteq \sum_{j \rightarrow_k \boldsymbol{x}} \boldsymbol{\alpha}_j \circ \boldsymbol{y}_j$

**Proof sketch:** The key to the proof, which we explore further in subsection 4.2, is the existence of a particular function $g_F$, strictly concave and symmetric with respect to $1/2$, which allows to rewrite the loss as:

$$F(x) \quad = \quad (-g_F)^\star(-x) \ , \tag{5}$$

where $\star$ denotes the (Legendre) convex conjugate. Convex conjugates have the property that their derivatives are inverses of each other. This property, along with (5), allows to simplify the computation of the derivatives of the loss, for any example $i$ in the inverse neighborhood of $j$:

$$\begin{aligned}
\frac{\partial F(y_{ic} H_c(\boldsymbol{x}_i))}{\partial \delta_j} &= y_{ic} y_{jc} F'(y_{ic} H_c(\boldsymbol{x}_i)) \qquad (6)\\
&= -y_{ic} y_{jc} ((-g_F)^\star)'(-y_{ic} H_c(\boldsymbol{x}_i))\\
&= -y_{ic} y_{jc} ((-g_F)')^{-1}(-y_{ic} H_c(\boldsymbol{x}_i))\\
&= -y_{ic} y_{jc} (1 - (g_F')^{-1}(-y_{ic} H_c(\boldsymbol{x}_i)))\\
&= -y_{ic} y_{jc} (g_F')^{-1}(y_{ic} H_c(\boldsymbol{x}_i))\\
&= -K_F w_i y_{ic} y_{jc} \ . \qquad (7)
\end{aligned}$$

Eq. (7) holds because we can also rewrite the weights update (Table 2) as:

$$w_i \quad \leftarrow \quad \frac{1}{K_F} (g_F')^{-1} \left( \delta_j y_{ic} y_{jc} + g_F'(K_F w_i) \right) \ , \tag{8}$$

where $(g_F')^{-1}$ is the inverse function of the first derivative of $g_F$, and $K_F$ is a normalizing constant: it is respectively $\ln(2), 1, 1/2, 1$ for A, B, C and D in Table 3. From (6), it also comes $\partial^2 F(y_{ic} H_c(\boldsymbol{x}_i))/\partial \delta_j^2 = F''(y_{ic} H_c(\boldsymbol{x}_i))$, where $F''$ denotes the second derivative. Considering the whole inverse neighborhood of $j$, the Newton-Raphson update for $\delta_j$ is (with $\eta(c, j) \doteq \sum_{i:j \rightarrow_k i} w_{ti} y_{ic} y_{jc}$ in $\mathrm{N}^3$):

$$\delta_j \quad \leftarrow \quad \lambda_F \times \frac{K_F \eta(c, j)}{\sum_{i:j \rightarrow_k i} F''(y_{ic} H_c(\boldsymbol{x}_i))} \ , \tag{9}$$

| crit | leveraging update, $\delta_j$ | weight update $g : w_i \leftarrow g(w_i, \delta_j, y_{ic}, y_{jc})$ |
|---|---|---|
| A | $\frac{4 \ln(2) \eta(c,j)}{n_j}$ | $\frac{w_i}{w_i \ln 2 + (1 - w_i \ln 2) \times \exp(\delta_j y_{ic} y_{jc})}$ |
| B | $\frac{4\eta(c,j)}{\ln^2(2) n_j}$ | $\frac{w_i}{w_i + (1 - w_i) \times 2^{\delta_j y_{ic} y_{jc}}}$ |
| C | $\frac{\eta(c,j)}{2n_j}$ | $1 - \frac{1 - w_i + \sqrt{w_i(2 - w_i)} \delta_j y_{ic} y_{jc}}{\sqrt{\begin{array}{l} 1 + \delta_{jc}^2 w_i (2 - w_i) \\ +2(1 - w_i)\sqrt{w_i(2 - w_i)} \delta_j y_{ic} y_{jc} \end{array}}}$ |
| D | $\frac{4\eta(c,j)}{n_j}$ | $\frac{1 + H\left(\delta_j y_{ic} y_{jc} + \frac{1 - 2w_i}{\mathrm{err}(w_i)}\right)}{2 + \left\lvert \delta_j y_{ic} y_{jc} + \frac{1 - 2w_i}{\mathrm{err}(w_i)} \right\rvert}$ |

**Table 2**. Leveraging and weight updates in $\mathrm{N}^3$ corresponding to each choice of calibrated loss in Table 1.

| crit | generator $g_F$ |
|---|---|
| A | $-x \ln x - (1 - x) \ln(1 - x)$ |
| B | $-x \log_2 x - (1 - x) \log_2(1 - x)$ |
| C | $\sqrt{x(1 - x)}$ |
| D | $\ln(2\mathrm{err}(x)) + 1 - 2\mathrm{err}(x)$ |

**Table 3**. Generators corresponding to calibrated losses in Table 1.

for learning rate $0 < \lambda_F \leq 1$. Matching this expression with the updates in Table 2 brings learning rate:

$$0 < \lambda_F = \frac{L_F \sum_{i:j \rightarrow_k i} F''(y_{ic} H_c(\boldsymbol{x}_i))}{K_F n_j} \leq \frac{L_F F''(0)}{K_F} = 1 \ ,$$

for each criteria A, B, C and D, where $L_F$ is respectively $4\ln(2), 4/\ln^2(2), 1/2, 4$, and $n_j \doteq |\{i : j \rightarrow_k i\}|$ in $\mathrm{N}^3$. The inequalities come from the fact that $F'' > 0$ and takes its maximum in 0 for all criteria. We then check that $F''(0) = K_F / L_F$ for A, B, C and D. □

### 4.2. A key to the properties of $\mathrm{N}^3$

The duality between real-valued classification and posterior estimation which stems from $f$ (See Section 3) is fundamental for the algorithmic and statistical properties of $\mathrm{N}^3$. To simplify the statement of results and proofs, it is convenient to make the parallel between our calibrated losses $F$ and functions elsewhere called permissible[1], that is, functions defined on $(0, 1)$, strictly concave, differentiable and symmetric with respect to $x = 1/2$. It can be shown that for any of our choices of $F$, there exists a permissible $g_F$, that we call a *generator*, for which the relationships (8) and (5) used in the proofsketch of Theorem 1 indeed hold. Furthermore, the generator is also useful to write the transfer function itself, as we have:

$$f(x) \quad = \quad (-g_F)'^{-1}(x) \ . \tag{10}$$

[1]The usual definitions are more restricted: for example the generator of calibrated Hinge loss would not be permissible in the definitions of [10, 6].

Table 3 provides the four generators corresponding to choices A, B, C and D. The permissible generator of the calibrated Hinge loss makes use of the error function:

$$\mathrm{err}(x) \quad \doteq \quad \min\{x, 1-x\} \ . \tag{11}$$

Permissible functions (as well as (11)) are used in losses that rely on posterior estimation rather than real-valued classification. Such losses are the cornerstone of decision-tree induction and other methods that directly fit posteriors [5]. Hence, (5) establishes a duality between the two kinds of losses, duality which appears as a watermark in various works [7, 11]. The writing of the weight update using $g_F$ in (8) is also extremely useful to simplify the proofs of the following Theorems. Finally, there is a synthetic writing for the weights, which sheds light on their interpretation: unraveling the weight update (8) and using (10), we obtain that $w_i$ satisfies:

$$w_i \quad \propto \quad 1 - f(y_{ic} H_c(\boldsymbol{x}_i)) \ . \tag{12}$$

Hence, weights and estimated posteriors are in opposite linear relationship. According to (12), examples "easier to classify" (receiving large estimated posteriors) receive small weight. This is a fundamental property of boosting algorithms, that progressively concentrate on the hardest examples.

## 5. ALGORITHMIC PROPERTIES OF N$^3$

The first result is a direct follow-up from Table 2.

**Lemma 1** *With choice D (calibrated Hinge loss),* N$^3$ *may be implemented using only rational arithmetic.*

**Comments on Lemma 1**: In the light of the boosting properties of N$^3$, this result is important in itself. Most existing boosting algorithms, including UNN, AdaBoost, Gentle AdaBoost and spawns [6, 11] make it necessary to tweak or clip the key numerical steps, including weights update or leveraging coefficients [2], at the possible expense of failing to meet boosting's convergence or accuracy. Rational arithmetic still requires significant computational resources with respect to floating point computation, but Lemma 1 shows that whenever these are accessible, formal boosting may be implemented *virtually without any loss in numerical precision*.

Let us now shift to the boosting result on N$^3$, which is stated under the following weak learning assumption:

> There exist constants $\gamma_u > 0, \gamma_n > 0$ such that at any iterations $c, t$ of N$^3$, index $j$ returned by WEO is such that $n_j > 0$ and the following holds: (i) $\frac{\sum_{i:j \to_k i} w_i}{n_j} \geq \frac{\gamma_u}{K_F}$, and (ii) $|\hat{p}_{\boldsymbol{w}}[y_{jc} \neq y_{ic} | j \to_k i] - 1/2| \geq \gamma_n$.

Requirement (ii) corresponds to the usual weak learning assumption of boosting: it postulates that the current *normalized* weights in the inverse neighborhood of example $j$ authorize a classification different from random by at least $\gamma_n$.

| | | $k$-NN | N$^3_{log}$ | N$^3_{binlog}$ | N$^3_{hinge}$ | N$^3_{mat}$ |
|---|---|---|---|---|---|---|
| ACC | $L1$ | 25.58 | 35.50 | 36.40 | 33.62 | 34.40 |
| | $L2$ | 25.90 | 33.97 | 35.44 | 32.87 | 33.55 |

**Table 4**. Top1 accuracy on CAL (64 splits, $L1$ or $L2$ normalization).

| | $k$-NN | N$^3_{log}$ | N$^3_{binlog}$ | SGD |
|---|---|---|---|---|
| Top1 ACC | 20.92 | 30.16 | 30.10 | 28.59 |
| Top5 ACC | 42.67 | 55.21 | 54.90 | 57.08 |

**Table 5**. Top5 accuracy on SUN (64 splits, $L1$ normalization).

Requirement (i) states that *unnormalized* weights must not be too small. This is a necessary condition as unnormalized weights of minute order do not necessary prevent (i) to be met, but would obviously impair the convergence of N$^3$ given the linear dependence of $\delta_j$ in the unnormalized weights. The following Theorem states that N$^3$ is a boosting algorithm.

**Theorem 2** *Suppose* N$^3$ *is ran for $T$ steps for each $c$, and that the weak learning assumption holds at each iteration of* N$^3$. *Denote $\mathcal{J}$ the whole multi-set of indexes returned by* WEO. *Then for any criterion A, B, C, D, the total calibrated risk does not exceed some $\varepsilon \leq F(0)$ provided:*

$$\sum_{j \in \mathcal{J}} n_j \quad = \quad \Omega\left(\frac{(C + |\varepsilon|)m}{\gamma_n^2 \gamma_u^2}\right) \ . \tag{13}$$

*Remark*: requirement $\varepsilon \leq F(0)$ comes from the fact that a leveraged NN with null leveraging vectors would make a total calibrated risk equal to $F(0)$.

## 6. EXPERIMENTAL EVALUATION

### 6.1. Settings: contenders, databases and features

We mainly report and discuss experiments of N$^3$ versus $k$-NN and support vector machines (SVM) implemented with Stochastic Gradient Descent SGD which represents the state of art among the classifiers on large scale datasets [12].

We abbreviate N$^3_{log}$, N$^3_{binlog}$, N$^3_{mat}$, N$^3_{hinge}$ the four flavors of N$^3$ corresponding respectively to rows A, B, C, D in Table 1. In N$^3$, WEO chooses the example with the largest current $\delta_j$.

The **datasets** used in this paper, Caltech256, and SUN are among the most challenging datasets publicly available for large scale image classification:

• Caltech256 [13] (CAL): This dataset is a collection of 30607 images of 256 object classes. Following classical evaluation, we use 30 images/class for training and the rest for testing.

• SUN [14] (SUN): This dataset is a collection of 108656 images divided into 397 scenes categories. We set the number of training images per class to 50 and we test on the remaining.

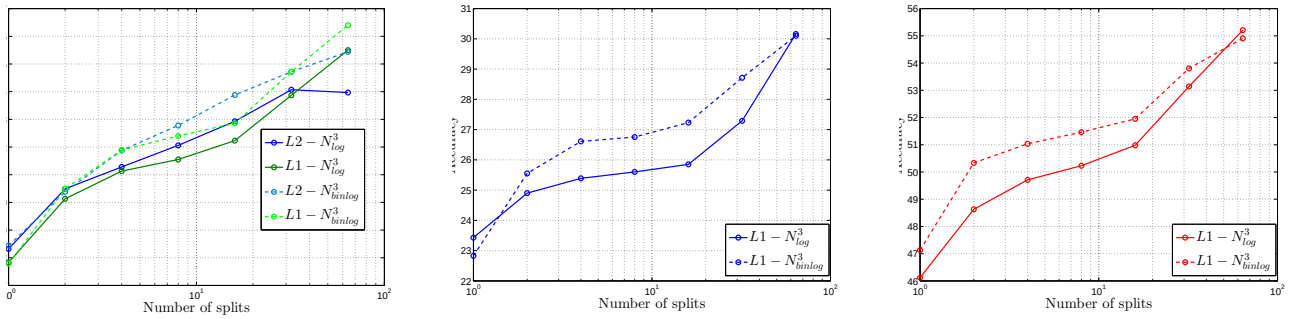


**Fig. 1**. The $x$-axis is the number of splits of the FV, on domains CAL (left) and SUN (center, right). The $y$-axis reports, using $L1$ or $L2$ normalization, the accuracy (left), top1 accuracy (center) and top5 accuracy (right) of $N^3$. Posteriors combined with the harmonic mean.

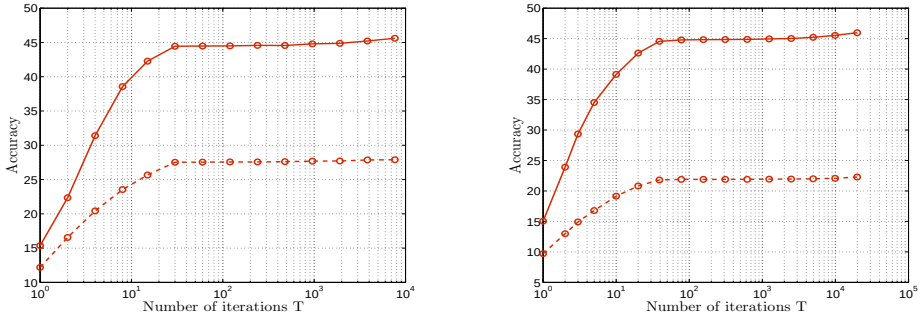


**Fig. 2**. Top1 and top5 accuracies (with 1 split) on CAL (left) and SUN (right) as a function of $T$.

We adopted for the **features** the Fisher vectors (FV) [15] encoding to represent images. Fisher Vector are computed over densely extracted SIFT descriptors ($FV_s$) and local color features ($FV_{sc}$), both projected with PCA in a subspace of dimension 64. Fisher Vectors are extracted using a vocabulary of 16 Gaussian and normalized separately for both channels and then combined by concatenating the two features vectors ($FV_{s+sc}$). This approach leads to to a 4K dimensional features vector.

To **compare** algorithms, we adopt the top1 and top5 accuracies (ACC), defined respectively as the proportion of examples that was correctly labelled and the proportion of those for which the correct class belongs to the top5 predicted patterns [3]. We also report processing times on a 2 X Intel Xeon E5-2687W 3,1GHz and analyse the convergence and the cost of $N^3$. But first, we propose a divide and conquer algorithm that optimizes classification using posteriors.

### 6.2. A divide and conquer algorithm to cope with the curse of dimensionality with low memory requirement

It is well known that NN classifiers suffer of the curse of dimensionality [16], hubs [17], so that the accuracy can decrease when increasing the size of descriptors. This may also affect $N^3$. FV are extremely powerful descriptors but they generate a space with about $4K$ dimension for 32 gaussians that could impair $N^3$ performance.

Our approach relies on nice property of minimizing classification-calibrated losses: we can easily compute the posteriors from the score using $N^3$ (see [18]). Thus, we propose a three step splitting method :

- split FV in a regular set of $n^* \in \{2, 4, 8, 16, 32, 64\}$ sub-descriptors and normalize with $L1$ or $L2$ norm;
- compute posteriors for each sub-vector (Table 1);
- combine these probabilities using a generalized average: arithmetic, geometric or harmonic.

### 6.3. Analysis on accuracy and convergence

First, figure 1 validates the divide and conquer approach, as increasing the number of splits on FV clearly improves performances. Also, as seen from the left plot, $L1$ normalization tends to outperform $L2$ normalization. The "optimal" number of splits (64) is then used in Table 4 which displays that $L1$ normalization of FV slightly improves classical $L2$ normalization. $N^3_{binlog}$ is also better than all other flavors of $N^3$, and overall all flavors of $N^3$ very significantly outperform $k$-NN.

We have also compared $N^3$ against SGD and $k$-NN on the SUN data set [14]. Results using $T = 50$ $iter$ for $N^3$ and 1000 $iter$ for SGD are displayed in Table 5. One sees that

$N^3$ significantly beats $k$-NN and approaches the accuracy of SGD. Note that memory requirement for $N^3$ is divided by the number of splitting (i.e. twice the number of Gaussian of the Fischer Vector).

Training time is very important for large scale data base processing. The training time of linear SGD is typically of order $\mathcal{O}(md)$. This results in hours of training reported by [15, 3, 19] where $m$ is the training data set size and $d$ is the features space dimension. On the other hand, NN classifiers become more efficient for huge data bases as reported by [19, 20, 3].

In fact, figure 2 shows the convergence of $N^3$ on CAL and SUN. One sees from the plots that the convergence of the Newton approach in $N^3$ is extremely fast and requires only few iterations — this is not the case for the non-Newton approach UNN [6], which requires a larger number of iterations. The fast convergence in $N^3$ results in sparse prototype selection ($T \ll m$), well adapted for large scale datasets, and suggests to choose $T$ as a function of the number of images in the corresponding class (inner loop of $N^3$), such as $T = \mathcal{O}(m/C)$. Hence, we end up with a complexity depending on $T \ll m$.

## 7. CONCLUSION

In this paper we have proposed a novel Newton-Raphson approach to boosting $k$-NN. We show that it is a boosting algorithm, with several key algorithmic and statistical properties. Experiments display that although accuracy results are similar to state of the art approaches like SGD , our $N^3$ requires memory divided by the number of Gaussian. This approach is suitable for very large scale image classification problems.

## 8. REFERENCES


[1] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro, "Pegasos: Primal estimated sub-gradient solver for svm," in *Proceedings of the 24th international conference on Machine learning*, New York, NY, USA, 2007, ICML '07, pp. 807–814, ACM.

[2] K. Ali, D. Hasler, and F. Fleuret, "Flowboost - appearance learning from sparsely annotated video," in *Proc. of the 24$^{th}$ IEEE CVPR*, 2011, pp. 1433–1440.

[3] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka, "Metric Learning for Large Scale Image Classification: Generalizing to New Classes at Near-Zero Cost," in *Procs of the 12 $^{th}$ ECCV*, 2012.

[4] K.-Q. Weinberger and L.-K. Saul, "Distance metric learning for large margin nearest neighbor classification," *JMLR*, vol. 10, pp. 207–244, 2009.

[5] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Springer, 1996.

[6] Richard Nock, Paolo Piro, Frank Nielsen, Wafa Bel Haj Ali, and Michel Barlaud, "Boosting k-nn for categorization of natural scenes," *IJCV*, vol. 100, pp. 294–314, 2012.

[7] P. Bartlett, M. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *JASA*, vol. 101, pp. 138–156, 2006.

[8] E. Vernet, R.-C. Williamson, and M. Reid, "Composite multiclass losses," in *NIPS*24*, 2011, pp. 1224–1232.

[9] J. Kivinen and M. Warmuth, "Relative loss bounds for multidimensional regression problems," *MLJ*, vol. 45, pp. 301–329, 2001.

[10] M.J. Kearns and Y. Mansour, "On the boosting ability of top-down decision tree learning algorithms," *JCSS*, vol. 58, pp. 109–128, 1999.

[11] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression : a Statistical View of Boosting," *AOS*, vol. 28, pp. 337–374, 2000.

[12] F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid, "Towards good practice in large-scale learning for image classification," in *Proc. of the 25$^{th}$ IEEE CVPR*, 2012 (to appear).

[13] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," Tech. Rep. 7694, California Institute of Technology, 2007.

[14] J. Xiao, J. Hays, K.-A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Proc. of the 23$^{rd}$ IEEE CVPR*, 2010, pp. 3485–3492.

[15] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Procs of the 11 $^{th}$ ECCV*, 2010, pp. 143–156.

[16] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft, "When is "nearest neighbor" meaningful?," in *Proc. of the 7$^{th}$ ICDT*, 1999, pp. 217–235.

[17] M. Radovanović, A. Nanopoulos, and M. Ivanović, "Hubs in space: Popular nearest neighbors in high-dimensional data," *JMLR*, vol. 11, pp. 2487–2531, 2010.

[18] R. D'Ambrosio, R. Nock, W. Bel Haj Ali, F. Nielsen, and M. Barlaud, "Boosting nearest neighbors for the efficient estimation of posteriors," in *Proc. of the 23 $^{rd}$ ECML-PKDD*, 2012 (to appear).

[19] Jia Deng, Alexander C. Berg, Kai Li, and Li Fei-Fei, "What does classifying more than 10,000 image categories tell us?," in *Procs of the 11 $^{th}$ ECCV*, 2010, pp. 71–84.

[20] Jason Weston, Samy Bengio, and Nicolas Usunier, "Wsabie: Scaling up to large vocabulary image annotation," in *IJCAI'11*, 2011, pp. 2764–2770.